UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/611,403	07/06/2000	Paul F. Ringseth	3382-56062	6514

7590 04/07/2004

Klarquist Sparkman Campbell Leigh & Whinston LLP
One World Trade Center
Suite 1600
121 SW Salmon Street
Portland, OR 97204-2988

EXAMINER	
WOOD, WILLIAM H	

ART UNIT	PAPER NUMBER
2124	8

DATE MAILED: 04/07/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 09/611,403	Applicant(s) RINGSETH ET AL.	
	Examiner William H. Wood	Art Unit 2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *10 October 2003*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-36* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-36* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Claims 1-36 are pending and have been examined.

Claim Rejections - 35 USC § 102

1. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

2. Claim 21 is rejected under 35 U.S.C. 102(a) as being anticipated by Richard Grimes, "Attribute Programming with Visual C++".

In regard to claim 21, Grimes disclosed the limitations:

- *In a computer system, a method of embedding debugging information in a definition language output file to facilitate debugging of an input file* (page 2, third paragraph under section "How is Attribute Programming Managed in Visual C++"), *the input file comprising constructs of definition language information embedded in programming language code* (pages 4-5, code block), *the method comprising:*
  - *receiving by a programming language compiler an input file, the input file comprising constructs of definition language information embedded in programming language code* (pages 4-5, code block);
  - *embedding by the programming language compiler debugging information in a definition language output file* (page 3, second and third paragraph

under figure), *the definition language output file for subsequent processing by a definition language compiler* (page 3, second and third paragraph under figure; page 2, third paragraph under section "How is Attribute Programming Managed in Visual C++"; MIDL), *whereby the embedded debugging information associates errors raised by the definition language compiler with locations of embedded definition language constructs in the input file to facilitate debugging of the input file* (page 3, second paragraph under figure).

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1-20 and 22-36 are rejected under 35 U.S.C. 103(a) as being unpatentable over Richard Grimes, "Attribute Programming with Visual C++" in view of Aho et al., "Compilers Principles, Techniques, and Tools" herein referred to as Grimes and Aho respectively.

In regard to claim 1, Grimes disclosed the limitations:

- *A computer readable medium having stored thereon a computer executable compiler system* (page 2, first paragraph of section "How is Attribute Programming Managed in Visual C++"; page 3, second paragraph under

figure; compiler system) *that performs semantic analysis of definition language information* (page 3, second paragraph under figure) *embedded in programming language code in a file* (page 2, first paragraph of section "How is Attribute Programming Managed in Visual C++", "interface" keyword; page 4-5, code block shows C++ and definition language code combined, notice "module" word), *the compiler system comprising:*

- *a file including programming language code having embedded therein definition language information* (page 2, first paragraph of section "How is Attribute Programming Managed in Visual C++", "interface" keyword; page 4-5, code block shows C++ and definition language code combined, notice "module" word);
- *output ... based at least in part upon semantics of the embedded definition language information* (page 3, second paragraph under figure).

Grimes did not explicitly state the limitations *a front end module that separates a file into plural tokens; a converter module that converts the plural tokens into an intermediate representation; and a back end module that produces output code from the intermediate is representation.* Aho demonstrated that it was known at the time of invention to develop compilers with a front end, a converter module and a back end (page 20, section "Front and Back Ends"). It would have been obvious to one of ordinary skill in the art at the time of invention to implement Grimes' system of C++ code and definition code with a compiler, which would generate executable code as found in Aho's teaching. This implementation would have been obvious because one of ordinary skill

in the art would be motivated to provide a mechanism, which would allow source code to produce meaningful executable code.

In regard to claim 2, Grimes and Aho further disclosed the limitation *wherein the intermediate representation includes a symbol table and a parse tree that unifies representation of the programming language code and the embedded definition language information* (Aho: pages 11 and 40-48).

In regard to claim 3, Grimes and Aho further disclosed the limitation *wherein the symbol table includes plural entries for symbol names for the programming language code, and wherein at least one of the plural entries has an associated list of definition language attributes* (Aho: page 11).

In regard to claim 4, Grimes and Aho further disclosed the limitation *further comprising a definition language attribute provider that modifies the intermediate representation based upon the semantics of the embedded definition language information* (Grimes: pages 2-3, paragraph spanning pages; page 3, figure shown).

In regard to claim 5, Grimes and Aho further disclosed the limitation *further comprising an error checker module that checks for lexical, syntactic, and semantic errors in the file* (Aho: page 11).

In regard to claim 6, Grimes disclosed the limitations:

- *In a computer system, a computer executable compiler system that creates a unified programming language ... from a file comprising a mix of programming language constructs and interface definition language constructs* (page 2-5), *the compiler system comprising:*
  - *a file comprising a mix of programming language constructs and interface definition language constructs* (page 4-5, code block);

Grimes did not explicitly state the limitations *interface definition language parse tree; a front end module that separates a file into plural tokens; and a converter module that converts the plural tokens into an intermediate representation comprising a symbol table and a parse tree, wherein the symbol table includes plural entries for symbol names for the programming language constructs, at least one of the plural entries having an associated list of interface definition language attributes, and wherein the parse tree unifies representation of the programming language constructs and the interface definition language constructs*. Aho demonstrated that it was known at the time of invention to develop compilers with a front end, a converter module, a back end, a symbol table, and a parse tree (page 1-24 and 40-48). It would have been obvious to one of ordinary skill in the art at the time of invention to implement Grimes' system of C++ code and definition code with a compiler, which would generate executable code as found in Aho's teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to provide a mechanism, which would allow source code to produce meaningful executable code. Finally, upon the above

combination, it can be seen that symbol tables (provide by Aho), would have entries that contain a list of attributes associated with interface definition language (provided by Grimes).

In regard to claim 7, Grimes and Aho disclosed the limitation *wherein the front end module recognizes a delimiting character that distinguishes interface definition language tokens from programming language tokens* (Grimes: page 4-5, code block demonstrates "module" preceded by "[", a delimiting character).

In regard to claim 8, Grimes and Aho further disclosed the limitation *further comprising an error checker module that performs lexical and syntactic checks on the file* (Aho: page 11).

In regard to claim 9, Grimes disclosed the limitations:

- *A computer readable medium having stored thereon a data structure representing a unified interface definition language ... for a file having a combination of programming language code and embedded interface definition language information* (page 2-5; notice Figure and code block)

Grimes did not explicitly state limitations concerning *programming language parse tree and symbol table*. Aho demonstrated that it was known at the time of invention to utilize parse trees and symbol tables (pages 11 and 40-48). It would have been obvious to one of ordinary skill in the art at the time of invention to implement Grime's interface

definition language / programming language compiler with symbol tables and parse trees as appropriate for compiling such a combination as found in Aho's teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to use common and well understood techniques for implementing compilers. Additionally the limitations below were discussed in previous claims:

- *a first data field storing data representing a symbol table that has plural entries, each of the plural entries corresponding to a symbol name for programming language code of a file having a combination of programming language code and embedded interface definition language information, at least one of the plural entries having an associated list of interface definition language attributes based upon the embedded interface definition language information* (page 11); *and*
- *a second data field storing data representing a parse tree, wherein the parse tree unifies representation of the programming language code and the embedded interface definition language information* (page 40-48).

In regard to claim 10, Grimes disclosed the limitations:

- *In a computer system, a method of creating a binary file from an input file that includes a mix of programming language constructs and definition language constructs* (page2, section "How is Attribute Programming Managed in Visual C++"; page 4-5, code block), *the method comprising:*

- *providing one or more input files, each input file comprising a mix of programming language constructs and definition language constructs* (page 4-5, code block);
- *upon user initiation at compile time, creating a binary file from the one or more input files, wherein the creation of the binary file comprises* (page 3, second paragraph under figure):
  - *with a compiler, converting the one or more input files into one or more output code files that include fragments of definition language information* (page 3, second paragraph under figure) *wherein the one o more output code files further include output computer-executable code based at least in part upon semantics of the definition language constructs* (page 3, second paragraph under figure)

Grimes did not explicitly teach *with a linker, generating a binary file from the one or more output code files.* Aho demonstrated that it was known at the time of invention to utilize linkage editors and loaders (page 19; section "Loaders and Link-Editors"). It would have been obvious to one of ordinary skill in the art at the time of invention to implement Grimes' system of compilation with a linkage editor as found in Aho's teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to provide functionality, which is commonly used to execute code/programs and provide a final executable version of a program/code.

In regard to claim 11, Grimes and Aho further disclosed the limitations wherein the generating comprises:

- *extracting the fragments of definition language information from the one or more output code files* (obvious from Aho and especially considering Grimes: page 3, third paragraph under the figure);
- *passing the extracted fragments to the compiler* (obvious from Aho and especially considering Grimes: page 3, third paragraph under the figure);
- *generating by the compiler an intermediate definition language file* (obvious from Aho and especially considering Grimes: page 3, third paragraph under the figure);
- *based upon the intermediate definition language file, generating by a definition language compiler a type library file* (obvious from Aho and especially considering Grimes: page 3, third paragraph under the figure); *and*
- *producing the binary file based upon the one or more output code files and the type library file* (obvious from Aho and especially considering Grimes: page 3, third paragraph under the figure).

In regard to claim 12, Grimes and Aho further disclosed the limitations wherein the producing comprises:

- *embedding the type library file into a first intermediate resource file* (obvious from Aho and especially considering Grimes: page 3, third paragraph under the figure);

- *with a resource tool, generating a second intermediate resource file* (obvious from Aho and especially considering Grimes: page 3, third paragraph under the figure);
- *with a resource file combiner, combining the second intermediate resource file with one or more related resource files into a combined resource file* (obvious from Aho and especially considering Grimes: page 3, third paragraph under the figure); *and*
- *producing the binary file based upon the one or more output code files and the combined resource file* (obvious from Aho and especially considering Grimes: page 3, third paragraph under the figure).

In regard to claim 13, Grimes disclosed the limitations:

- *In a computer system, a method of deriving semantic meaning from definition language information embedded in programming language code in a file* (pages 2-3, section "How is Attribute Programming Managed in Visual C++", including the figure; pages 4-5, block of code), *the method comprising:*
  - *a file including definition language information embedded in programming language code* (pages 4-5, block of code)
  - *representation based at least in part upon semantics of the embedded definition language information* (page 3, figure)

Grimes did not explicitly teach *separating a file into plural tokens; converting the plural tokens into an intermediate representation; and generating output code from the*

intermediate representation. Aho demonstrated that it was known at the time of invention to provide compilers, which utilize separating files into tokens, converting tokens to an intermediate representation, and generating output code from an intermediate representation (pages 4-15; page 4 mentions tokens, page 12 mentions the intermediate representation). It would have been obvious to one of ordinary skill in the art at the time of invention to implement Grimes' programming language code embedded with definition language information compiler with the techniques found in Aho's teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to develop a compiler based upon the well understood compiler theories and constructions taught by Aho for the purpose of building compilers.

In regard to claim 14, Grimes and Aho further disclosed the limitations wherein the converting comprises:

- *building a symbol table having plural entries for symbol names for the programming language code, at least one of the plural entries having an associated list of definition language attributes based upon the embedded definition language information* (Aho: page 11)*; and*
- *building a parse tree that unifies representation of the programming language code and the embedded definition language information* (Aho: page 40-48, section 2.4)

In regard to claim 15, Grimes and Aho further disclosed the limitation *further comprising modifying the intermediate representation by a definition language attribute provider based upon the semantics of the embedded definition language information* (Grimes: page 3, figure shows attribute provider; page 2-3, paragraph spanning the pages describes the operations of attribute providers).

In regard to claim 16, Grimes disclosed the limitations:

- *A computer readable medium having stored thereon instructions for performing a method of creating a unified programming language* (page 3, figure; page 4-5, code block) *... a file that includes definition language information embedded in programming language code* (page 4-5, code block), *the method comprising:*
  - *a file including definition language information embedded in programming language code* (page 4-5, code block)

Grimes did not explicitly state *definition language parse tree; separating a file into plural tokens; building a symbol table having plural entries for symbol names for the programming language code, at least one of the plural entries having an associated list of definition language attributes based upon the embedded definition language information; and building a parse tree that unifies representation of the embedded definition language information and the programming language code.* Aho demonstrated that it was known at the time of invention to utilize compilers with various features, including: parse trees, breaking files into a plurality of tokens, building symbol

tables (pages 10-11, 20, and 40-48). It would have been obvious to one of ordinary skill in the art at the time of invention to implement Grimes' definition language information and unified programming language compiler system with the tools necessary for compiler's to function as found in Aho's teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to allow the compiler of Grimes to function as is commonly known for compilers. Upon see the obviousness of the two references in combination, one of ordinary skill in the art would also clearly see the limitations *symbol table having plural entries for symbol names for the programming language code, at least one of the plural entries having an associated list of definition language attributes based upon the embedded definition language information* (symbol table of Grimes would include both programming language and definition language in order for the compiler to correctly track identifiers that may occur) and *parse tree that unifies representation of the embedded definition language information and the programming language code* (parse tree in Grimes in order to correctly processing the tokens of a system that has programming language and definition language constructs).

In regard to claim 17, Grimes and Aho further disclosed the limitation *wherein the separating comprises recognizing a delimiting character that distinguishes definition language tokens from programming language tokens* (Grimes: page 4-5, code block; note "[" near the word "module").

In regard to claim 18, Grimes disclosed the limitations:

- *A computer readable medium having stored thereon a computer executable compiler system that checks for errors in a file comprising a mix of definition language information and programming language code* (page 2-5; figure and code block), *the compiler system comprising:*

Grimes did not explicitly state the limitations *a front end module that separates a file into plural tokens* and checking for errors; *a converter module that converts the plural tokens into an intermediate representation* and checking for errors (typically part of the front end); *and a back end module that produces output code from the intermediate is representation.* Aho demonstrated that it was known at the time of invention to develop compilers with a front end, a converter module and a back end (page 20, section "Front and Back Ends" and additional details of functions performed by those elements of a compiler are found throughout chapter 1, pages 1-24). It would have been obvious to one of ordinary skill in the art at the time of invention to implement Grimes' system of C++ code and definition code with a compiler, which would generate executable code as found in Aho's teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to provide a mechanism, which would allow source code to produce meaningful executable code.

In regard to claim 19, Grimes and Aho did not explicitly state *wherein the converter module further checks for semantic errors between the definition language information and the programming language code.* Aho demonstrated that it was known at the time

of invention to check for errors at various phases (page 11, section "Error Detection and Reporting"). Grimes demonstrated it was known to implement compilers with both definition language information and programming language information. It would have been obvious to one of ordinary skill in the art at the time of invention to implement the Grimes Aho compiler system with error checking between the definition language information and the programming code as suggested by their own teaching. The converter modules would check for errors just like all other phases/modules/sections. Furthermore, semantic errors are related to the converter module as it is related to language representation to begin with. This implementation would have been obvious because one of ordinary skill in the art would be motivated to use known compiler techniques and reduce errors.

In regard to claim 20, Grimes disclosed the limitations:

- *In a computer system having a programming language compiler that generates output code based upon programming language source code* (page 2-3, section "How is Attribute Programming Managed in Visual C++"), *the programming language compiler including a compiler state* (page 3, Figure shown), *an improvement comprising:*
  - *modifying the programming language compiler to recognize constructs of interface definition language information embedded within programming language source code* (pages 4-5, block of code showing definition language embedded);

- *modifying the programming language compiler to expose the compiler state to one or more interface definition language attribute providers* (page 3, figure shown);
- modifying the programming language compiler to allow manipulation of the elements of the compiler by the one or more interface definition language attribute providers based upon the semantics of the embedded interface definition language information (page 3, figure shown; pages 2-3, paragraph spanning the pages)

Grimes did not explicitly state the limitations of a symbol table and a parse tree. Aho demonstrated that it was known at the time of invention to develop compilers with a symbol table and a parse tree (page 10, 11, 40-48). It would have been obvious to one of ordinary skill in the art at the time of invention to implement Grimes' system of embedded definition code within a programming language with a compiler, which would generate executable code as found in Aho's teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated by the commonly understood techniques of allowing source code to produce meaningful executable code.

Claim 22

Grimes and **Aho** disclosed the compiler system of claim 1 wherein the backend module also produces output definition language information in an output file that includes the output computer-executable code (**Grimes**: page 2, third paragraph under "How is

Attribute Programming Managed in Visual C++?"; the sentence, "This indicates that IDL should be generated from the attributes and placed into the compiled .obj file.").

Claim 23

Grimes and **Aho** disclosed the compiler system of claim 1 wherein the backend module also produces output definition language information in a separate output file from the output computer-executable code (**Grimes**: page 2, third paragraph under "How is Attribute Programming Managed in Visual C++?"; the sentence, "The preview comes with a tool called Idlgen that appears to do the dual step of generating an IDL file from the .obj file and then...").

Claim 24

Grimes and **Aho** disclosed the compiler system of claim 1 wherein the output computer-executable code is computer-executable for a real processor (**Aho**: page 1, last paragraph to page 2, top paragraph).

Claim 25

Grimes and **Aho** did not explicitly state the compiler system of claim 1 wherein the output computer-executable code is computer-executable instructions for a virtual processor. Official Notice is taken that it was known at the time of invention to provide compilers for virtual processors. It would have been obvious to one of ordinary skill in the art at the time of invention to implement the compiler of **Grimes** and **Aho** with

compiling for a virtual machine/processor. This implementation would have been obvious because one of ordinary skill in the art would be motivated to provide compiling technology to as many possible compiler implementations and thus improve product usability and flexibility.

Claim 26

Grimes and **Aho** disclosed the compiler system of claim 1 wherein the programming language code is in C++ and wherein the embedded definition language information includes IDL constructs (**Grimes**: page 2, second and third paragraphs under "How is Attribute Programming Managed in Visual C++?")

Claims 27-36

The limitations of claims 27-36 correspond to claims 22-26 and are therefore rejected in the same manner.

Response to Arguments

5. Applicant's arguments filed 10 October 2003 have been fully considered but they are not persuasive. Applicant argued: [1)] **Grimes** did not disclose producing output computer-executable code based upon the semantics of the definition language information; [2)] Applicant argued **Grimes** and **Aho** failed disclose the limitations of claims 6, 9, 16 and 20; and [3)] **Grimes** did not disclose embedding debugging information in a definition language output file.

As to an initial matter, it is important to note **Grimes** *disclosed* the limitations of Applicant's claimed invention. The mutual understanding of **Grimes** is that the reference discussed the features of a compiler in existence in 1998. **Grimes** further states, "I will take this opportunity to give you some pointers as to what you can do with *this preview* and to determine what attribute programming will be like in the future" (page 1, second paragraph; emphasis added, meaning the this *compiler*). **Grimes** discussed the then existing compiler and then discussed the future of attribute programming. The fact that, a reference may or may not disclose more than Applicant's claimed invention has no bearing on whether Applicant's claimed invention reads upon the cited prior art.

As to the first issue, Applicant argued **Grimes** did not disclose producing output computer-executable code based upon the semantics of the definition language information. Applicant, further stated **Grimes** taught away from such an implementation. Both arguments are incorrect. **Grimes** disclosed output code based upon semantics of definition language information (page 3, second paragraph under figure; page 2, first through third paragraphs under "How is Attribute Programming Managed in Visual C++?"). Applicant even admits to this (Remarks filed 10 October 2003, page 13, line 7 to page 14, top. There does appear to be any support for Applicant's argument. Applicant simply asserts the limitation isn't there. **Grimes** specifically states, "The technical preview integrates interface (IDL) definition into the C++ code that uses it. To do this Microsoft has added the interface keyword to the C++ language" (page 2, first paragraph under "How is Attribute Programming Managed in

Visual C++?"). This alone indicates outputting computer-executable code based upon the *semantics of definition language information*, by the fact that the compiler will compile said code and output a file clearly using IDL inforamation. Furthermore, **Grimes** does not teach away from the claimed invention. **Grimes** states, "Defining interfaces like this within C++ is a little restrictive. I know many developers who find the current, separate IDL step a little confusing ...". **Grimes** clearly teaches *toward* both implementations. There are reasons for and against taking mixing the definition language and the programming language.

As to the second issue, Applicant argued **Grimes** and **Aho** failed disclose the limitations of claims 6, 9, 16 and 20, yet offers only the same argument as above for claims 1, 10, 13 and 18. Thus, Applicant is referred to the above issue and the previous rejection, which maps the claim language to the cited references.

As to the third issue, Applicant argued **Grimes** did not disclose embedding debugging information in a definition language output file. This is inaccurate according to paragraphs 2 and 3 under the figure on page 3 of **Grimes**. The broadest reasonable interpretation of claim 21 does not force a particular definition of "a definition language output file". The project's PDB file is read on by claim 21 and so is the attributes being placed in the .obj file. The cited portions of **Grimes** state debugging information being placed in a file by the compiler.

The above issues are believed to respond to all of Applicant's concerns. All other similar claims and dependent claims are believed to be addressed by the above issues as well. Thus, the claims remain rejected.

Conclusion

6. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Correspondence Information

Any inquiry concerning this communication or earlier communications from the examiner should be directed to William H. Wood whose telephone number is (703)305-3305. The examiner can normally be reached 7:30am - 5:00pm Monday thru Thursday and 7:30am - 4:00pm every other Friday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703)305-9662. The fax phone numbers for the organization where this application or proceeding is assigned are (703)746-7239 for regular communications and (703)746-7238 for After Final communications.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703)305-3900.

William H. Wood
March 31, 2004

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100